Requirements Engineering 1: "Introduction"

Steve Easterbrook

8/20/97

Outline

Requirements in the development lifecycle

The essential requirements process

Importance of requirements engineering

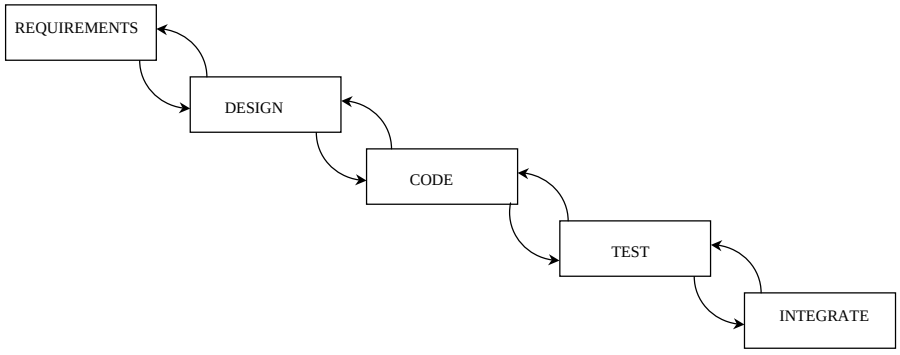
Specifications: types and audiences

An introduction to systems analysis

Lifecycle Models

- **Waterfall (and V-model)**
- **Rapid Prototyping**
- **Phased Models:**
  - **Incremental Development**
  - **Evolutionary Development**
- **Risk Management (Spiral Model)**
- **Domain Engineering model**

Waterfall Model



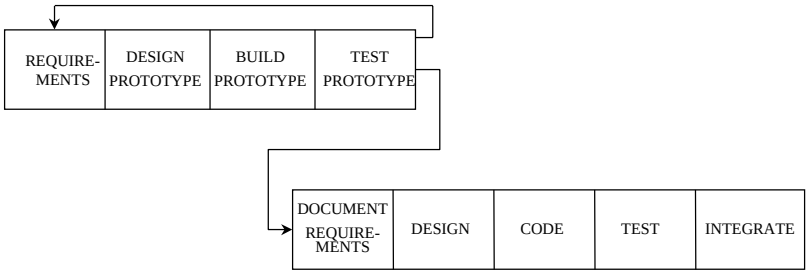
Source: *Adapted from Dorfman, 1997, p7*

Requirements and Waterfalls

- **Waterfall model describes a process of stepwise refinement**
- **Requirements Analysis is the second step:**
  - **1. Market Analysis (or business planning, or systems engineering)**
  - **2. Requirements analysis**
  - **3. Design**
  - **etc.**
- **Problems:**
  - **Waterfall model takes a static view of requirements, ignores volatility**
  - **Lack of user involvement once specification is written**
  - **Unrealistic separation of specification from design**
  - **Doesn't accommodate prototyping, reuse, etc.**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p29*

Prototyping lifecycle



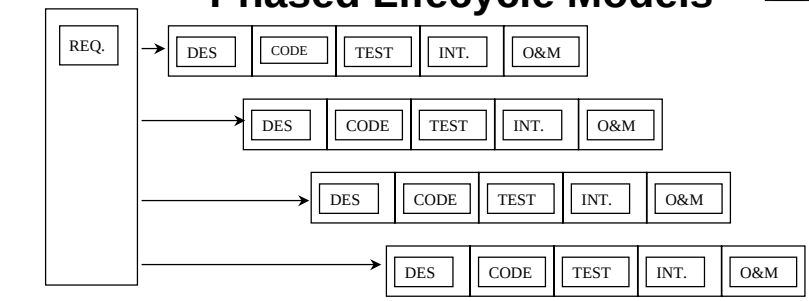
Source: *Adapted from Dorfman, 1997, p9*

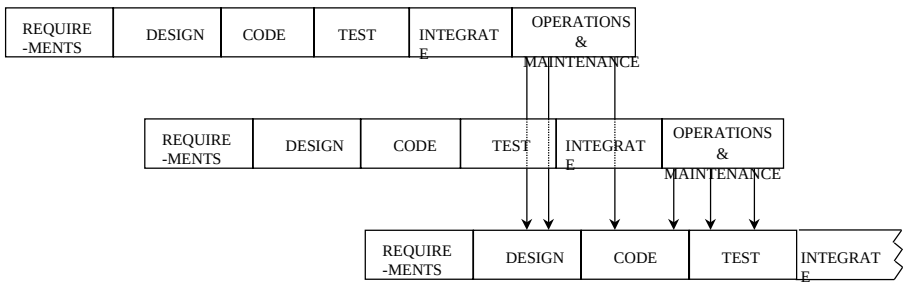
Requirements and Prototyping

- **Prototyping is used for:**
  - **understanding the requirements for the user interface**
  - **examining feasibility of a proposed design approach**
  - **exploring system performance issues**
- **Requirements Engineering**
  - **Elicitation by involving the user in experimental use of prototypes**
  - **Analysis by analyzing structure and behavior of the prototype**
  - **The prototype acts as a formal specification**
- **Problems:**
  - **users treat the prototype as the solution**
  - **a prototype is only a partial specification**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p30*

Phased Lifecycle Models





Source: *Adapted from Dorfman, 1997, p10*

Requirements in the Spiral Model

- **Spiral model is a risk management model**
- **For each iteration:**
  - **plan next phases; determine objectives & constraints; evaluate alternatives; resolve risks; develop product**
- **Includes as Requirements processes:**
  - **Requirements risk analysis (using simulation and prototyping)**
  - **Planning for design**

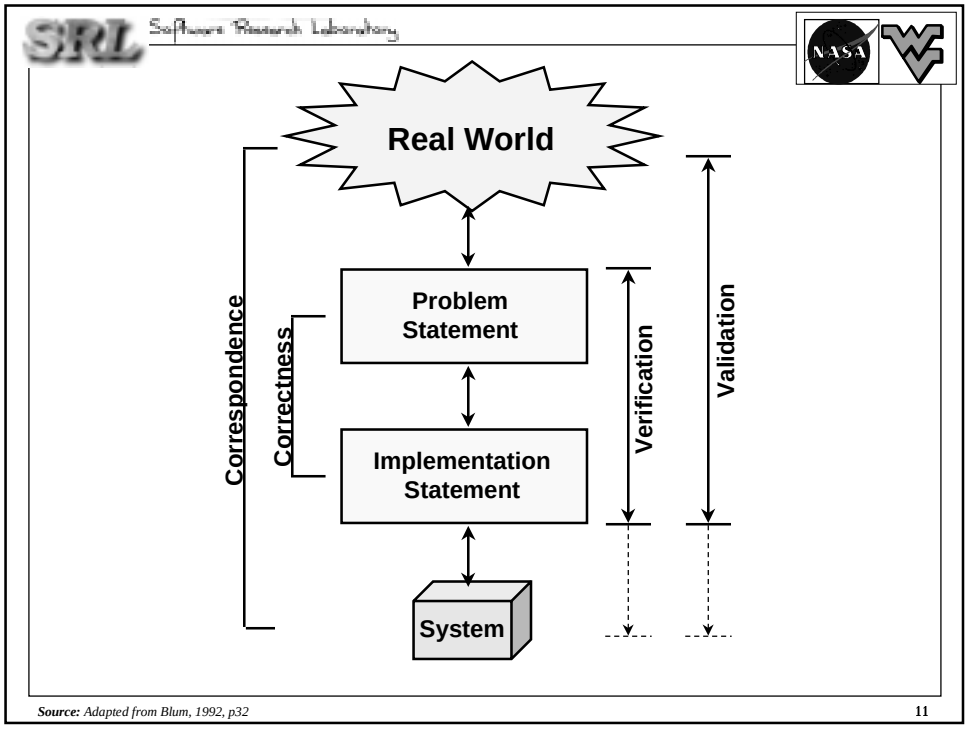
  (these reduce the risk that requirements process has to be repeated because requirements cannot be met)
- **Problems:**
  - **Spiral model cannot cope with unforeseen changes during development (e.g. emergence of new business objectives)**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p30*

Requirements and Domain Engineering

- **The approach:**
  - **Recognizes that requirements is not a one-off activity**
  - **Exploits similarity between applications in the same problem domain**
  - **Facilitates reuse**
- **Requirements**
  - **Requirements engineering follows domain analysis**
  - **Domain analysis looks at commonalities between applications**
  - **'problem understanding' is reduced to a process of mapping between the domain model and the needs of the specific application**
  - **'specification' consists of selection of an appropriate component from a library of reusable analysis components**
- **Problems**
  - **Domain analysis is an expensive effort, with a deferred payoff**
  - **Deadlines for application development tend to dominate**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p35*



Source: *Adapted from Blum, 1992, p32*

The essential requirements process

- **Understand the problem**
  - **elicitation, requirements acquisition, etc.**
- **Formally describe the problem**
  - **specification, modelling, etc.**
- **Attain agreement on the nature of the problem**
  - **validation, conflict resolution, negotiation**
  - **requirements management - maintain the agreement!**

Source: *Adapted from Loucopoulos & Karakostas, 1995, p20*

Nature of the requirements process

- Analysis problems have ill-defined boundaries
- Requirements are found in organizational contexts (hence prone to conflict)
- Solutions to analysis problems are artificial
- Analysis problems are dynamic
- Tackling analysis requires interdisciplinary knowledge and skill

Source: *Adapted from Loucopoulos & Karakostas, 1995, p14*

Importance of Requirements

- The Engineering argument
  - Engineering is about developing solutions to problems
  - A good solution can only be developed if the engineer has a solid understanding of the problem
- The Economic argument
  - Errors cost more the longer they go undetected
  - Cost of correcting a requirements error is 100 times greater in the maintenance phase than in the requirements phase
- The “don’t be a lemming” argument
  - Failure to understand and manage requirements is the biggest single cause of cost and schedule over-runs
- The Safety Argument...

Requirements and Safety

- **Fault analysis on Voyager (87 faults) & Galileo (122):**
  - **Safety-related interface faults overwhelmingly caused by communication errors between development teams (93%, 72%)**
  - **Functional faults primarily caused by errors in recognising (understanding) requirements (62%, 79%)**
  - **Safety-related tend to be errors in specifying requirements, while non-safety tend to be errors in implementing requirements.**
- **Safety-related software errors arose most often from inadequate or misunderstood requirements**
- **Difficulties with requirements is the key root cause of safety-related errors discovered in system testing**

(Source: Lutz, 1993)

Specification - Roles

- **Requirements Specification**
  - **details the concerns of the customers and users**
  - **defines functions to be performed, and constraints**
- **System Specification**
  - **Defines the system boundary and the interactions between the system and its environment**
  - **a "black box" view**
- **Architectural Design specification**
  - **identifies the major subsystems, and interactions between them**
  - **allocates functional requirements to the subsystems**
- **Detailed design specification**
  - **describes the details of the decomposed components of the system**

Audience for Requirements Specs

- **Users, purchasers**
  - **Most interested in system requirements**
  - **Not generally interested in detailed software requirements**
- **Systems Analysts, Requirements Analysts**
  - **author various specs that inter-relate**
- **Developers, Programmers,…**
  - **Have to implement the requirements**
- **Testers**
  - **Need to determine that requirements have been met**
- **Project Managers**
  - **Need to measure and control the analysis and development processes**

Secondary Stakeholders

- **Standards committees**
- **Safety boards**
- **The general public**
- **Congress (& other tiers of government)**
  - **even if they are not the purchasers or users**
- **Marketing Department**
- **Senior Management**

Three roles for specifications

- **A contract**
  - **specifies a job to be done**
  - **acts as basis for judging completion of the job (and hence payment!)**
- **A communication medium**
  - **Conveys an understanding of the domain**
  - **Passes information between different teams in the software development process**
- **A statement of commitment**

Definitions

- **Method**
  - **"A systematic way of working to achieve a desired result"**
  - **Requirements methods usually provide a set of notations, and some heuristics for how to use them**
- **Technique**
  - **A recipe for obtaining a certain result.**
  - **Techniques prescribe a way of working in detail (algorithmically)**
- **Notation**
  - **A systematic way to represent something**
  - **May be linguistic or graphical**
  - **May be formal or informal**
- **Methodology - "the study of methods"**

Source: *Adapted from Wieringa, 1996, p5*

Systems...

What is a system?

- **An actual or possible part of reality that can be observed or interact with it's environment**
  - **cars, cities, houseplants, rocks, spacecraft, buildings...**
  - **operating systems, DBMS, the internet, an organization**
  - **NOT: numbers, truth values, letters.**
- **A "closed system" is a system that does not interact with it's environment**
  - **There are no closed systems in reality!**
- **Systems might have no physical existence**
  - **The only physical manifestations are symbolic or analogical representations of the system**
  - **Such systems are social constructs: they exist because we agree on ways to observe them**

Source: *Adapted from Wieringa, 1996, p10*

Types of System

- **Natural Systems**
  - **E.g. ecosystems, weather, water cycle, the human body, bee colony, ...**
- **Abstract Systems**
  - **E.g. set of mathematical equations, computer programs, etc**
- **Designed Systems**
  - **E.g. cars, planes, buildings, interstates, telephones, the internet, ...**
- **Human Activity Systems**
  - **E.g. Organizations, markets, clubs, …**
- **… and any system may be hard or soft**
  - **soft - cannot represent the system precisely**
  - **hard - the system is precise, well-defined and quantifiable.**

Source: *Adapted from Carter et. al., 1988, p12*

System Boundary

- **System Environment:**
  - **the part of the world with which the system can interact**
  - **every system has an environment**
  - **the environment is itself a system**
  - **Distinction between system and environment depends on your viewpoint**
  - **System interface (or boundary) is the set of all possible interactions between system and environment**
- **Choosing the boundary**
  - **Some choices make more sense than others**
  - **Choice should be made to maximize modularity**
- **Examples:**
  - **Telephone system - include: switches, phone lines, handsets, users, accounts?**
  - **Desktop computer - do you include the peripherals?**
  - **Flight control system - do you include the ground control?**

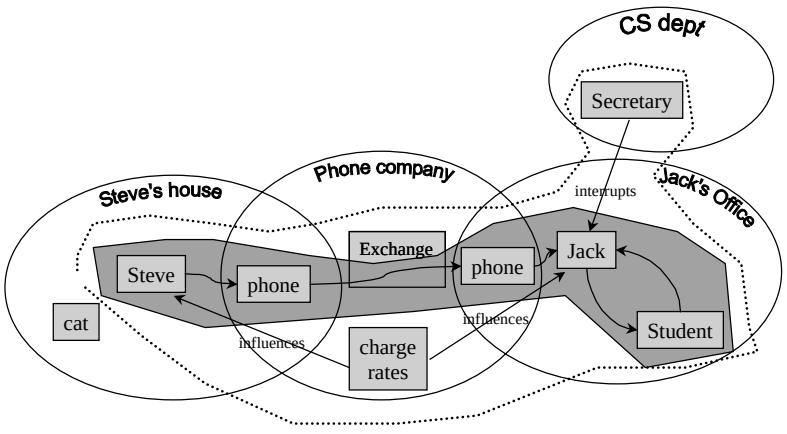
Source: *Adapted from Wieringa, 1996, p11-12*

Boundary Drawing

- **Exclude:**
  - **Components that have no functional effect on the system (relevant to it's purpose)**
  - **Components that influence the system but cannot be influenced or controlled by the system**
- **Include:**
  - **Items that can be strongly influenced or controlled by the system**
- **Balance:**
  - **between totally open and totally closed systems**
- **Modularity:**
  - **try to either include or exclude complete clusters of interactions to preserve modularity**

Source: *Adapted from Carter et. al., 1988, p6-7*

Example System Boundary



Source: *Adapted from Carter et. al., 1988, p6*

Achieving Modularity

- **Guidelines:**
  - **does the system have an underlying idea that can be described in one or two sentences?**
  - **Interaction among system components should be greater than interaction between the system and it's environment**
  - **Changes within a system should cause minimal changes outside**
  - **More 'energy' is required to transfer something across the system boundary than within the system boundary**
  - **The system boundary should 'divide nature at its joints'**
- **Choose the boundary that:**
  - **increases regularities in the behaviour of the system**
  - **simplifies the system behavior**

Source: *Adapted from Wieringa, 1996, p12*

Control

- **Control holds a system together**
- **structure and control are closely connected**

- **Self-maintaining causal network**
  - **a self-enhancing process: growth of the internet**
  - **a self-confirming process: visibility of a footpath**
  - **a self-limiting process: pricing of commodities**
- **Purposive Control**
  - **control sub-systems directed towards achieving a goal**
  - **"purpose without choice"**
- **Purposeful Control**
  - **special arrangements for decision making and control**
  - **Free choice among competing alternatives**
  - **"purpose with choice"**

Source: *Adapted from Carter et. al., 1988, p16*

Controllers

- **Purposive and purposeful systems need a controller**
- **Adaptive Control**
  - **A feedback loop is created to monitor the system and adjust the control accordingly**
- **Non-adaptive Control**
  - **System is set up reliably in advance, so no monitoring is needed**
  - **(sometimes called "open loop" or "feed-forward systems")**
- **Both types require:**
  - **knowledge about the system (a model?)**
  - **awareness of the needs the controller is trying to satisfy**

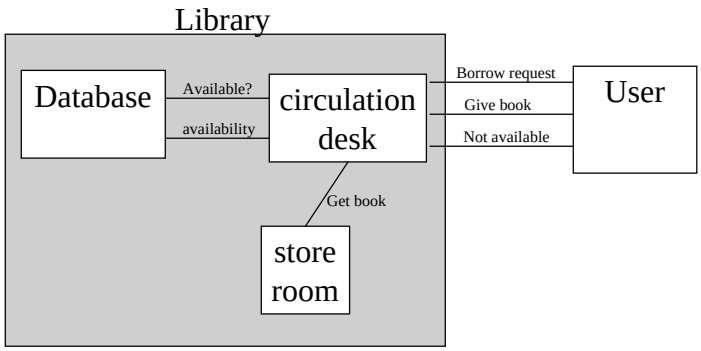
Source: *Adapted from Carter et. al., 1988, p17*

System Structure

- **Subsystems…**
  - **are systems too!**
  - **A system is an organised collection of subsystems acting as a whole**
  - **Subsystem boundaries should be chosen so that subsystems are modular**
- **An *Aspect* of a system**
  - **is a restricted subset of the interactions between its subsystems**
  - **E.g. for a car: all interactions to do with safety**
  - **(note fluidity between safety as an aspect, and safety as a subsystem)**
- **Visibility**
  - **Interactions between subsystems only are *internal* to the system**
  - **Interactions between subsystems and the environment are *external***
  - **Engineers usually try to hide internal interactions**
  - **For social systems, the internal interactions can be hidden too.**

Source: *Adapted from Wieringa, 1996, p13*

Example



Source: *Adapted from Wieringa, 1996, p14*

System State

- **State**
  - **a system will have memory of its past interactions, i.e. 'state'**
  - **the state space is the collection of all possible states**
- **Discrete vs continuous**
  - **a discrete system is a system whose states can be represented using natural numbers**
  - **a continuous system is one in which state can only be represented using real numbers**
  - **in a hybrid system some aspects of state can be represented using natural numbers**
- **Observability**
  - **the state space is defined in terms of the observable behavior**
  - **the perspective of the observer determines which states are observable**

Source: *Adapted from Wieringa, 1996, p16-17*

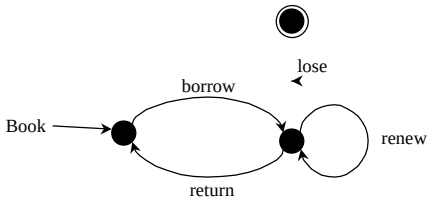
System transactions

- **Any observable interaction**
  - **starts in one state (the 'initial state')**
  - **ends in another (the 'final state')**
  - **may pass through a number of intermediate states**
- **A transaction is...**
  - **...any interaction that has no intermediate states**
  - **...atomic**
  - **(but atomicity depends on the level of detail being observed)**
- **If atomicity is required at one level of detail, it must be implemented at the lower level**
  - **e.g. through rollback and commitment**

Source: *Adapted from Wieringa, 1996, p17-18*

System Behavior

- **Behavior of discrete systems can be modelled using transition diagrams:**



- **Behavior of continuous systems cannot be modeled in this way**
  - **(but it can be abstracted into a discrete model)**

Source: *Adapted from Wieringa, 1996, p19-20*

System Properties

- **A system property**
  - **is an aspect of system behavior**
  - **(often referred to as 'attributes' or 'quality attributes')**
- **Specifying properties:**
  - **A property is specified behaviorally if an experiment has been specified that will tell us unambiguously whether the system has the property**
  - **A property is specified non-behaviorally if no such experiment has been identified**
  - **(sometimes referred to as functional / non-functional)**
  - **Presence of non-behavioral properties is a subjective (consensual) decision**
- **Proxies**
  - **If a property cannot be specified behaviorally, it may be possible to use a different property to indicate the presence of the first property**
  - **E.g. 'easy to learn', 'easy to use' as proxies for 'user friendly'**

Source: *Adapted from Wieringa, 1996, p20-21*

Next Week

- **The systems engineering process**
- **What is a requirement?**
- **SRS Evaluation exercise**

References

- Dorfman, M. “Requirements Engineering”. In Thayer, R. H and Dorfman, M. (eds.) “Software Requirements Engineering, Second Edition”. IEEE Computer Society Press, 1997, p7-22
- Loucopoulos, P. and Karakostas, V. “System Requirements Engineering”. McGraw Hill, 1995.
- Blum, B. “Software Engineering: A Holistic View”. Oxford University Press, 1992.
- Lutz, R. R., “Analyzing Software Requirements Errors in Safety-Critical Embedded Systems”. Proceedings of the IEEE International Symposium on Requirements Engineering, San Diego, CA, 4-6 January, 1993, p126-133
- Wieringa, R. J. “Requirements Engineering: Frameworks for Understanding”. Wiley, 1996.
- Carter, R., Martin, J., Mayblin, B., and Munday, M. “Systems, Management and Change: A graphic Guide”. Paul Chapman Publishing, 1988.